APPLICATION FOR UNITED STATES LETTERS PATENT

FOR

APPARATUS AND METHODS TO AVOID FLOATING POINT CONTROL INSTRUCTIONS IN FLOATING POINT TO INTEGER CONVERSION

Inventors: Qi ZHANG
Jianhui LI
Orna ETZION

Intel Reference No.: P17857
EPLC Reference No.: P-6215-US
Prepared By: Danny Kaufman
Eitan, Pearl, Latzer & Cohen Zedek, LLP.

APPARATUS AND METHODS TO AVOID FLOATING POINT CONTROL INSTRUCTIONS IN FLOATING POINT TO INTEGER CONVERSION

BACKGROUND OF THE INVENTION

[0001] ANSI/IEEE standard 754-1985 for binary floating-point arithmetic defines four rounding modes to affect all arithmetic operations except comparison and remainder: round to nearest, round toward negative infinity, round toward positive infinity, and round to zero.

[0002] In some processor architectures, prior to execution of an instruction to convert a floating point number to an integer using a specific rounding mode, the processing unit needs to be set to that rounding mode. This is achieved by reading and storing the current rounding mode, setting the processor architecture to the desired rounding mode, performing the conversion and setting processor architecture to the stored rounding mode. The instructions of setting the desired rounding mode and setting the stored rounding mode are examples of floating point control instructions.

[0003] Execution of a floating point control instruction may be time consuming and may degrade the processor architecture performance.

[0004] If a binary code associated with a first processor architecture having a first instruction set is to be executed by a second processor architecture having a second instruction set, a binary translation module may translate the source binary code into a target binary code associated with the second processor architecture. The results produced by executing the target binary code on a processor that complies with the target architecture are substantially the same as those produced by executing the source binary code on a processor that complies with the source architecture.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Embodiments of the invention are illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like reference numerals indicate corresponding, analogous or similar elements, and in which:

[0006] FIG. 1 shows a binary-code translation module according to some embodiments of the invention;

[0007] FIG. 2 is a flowchart illustration of an exemplary method to be implemented in a binary code translation module for translating a portion of a source binary code to a portion of a target binary code, according to some embodiments of the invention;

[0008] FIGS. 3, 4, 5A and 5B are flowchart illustrations of exemplary instruction sequences to be generated by a binary code translation module, according to some embodiments of the invention;

[0009] FIG. 6 is a block diagram of an exemplary apparatus according to some embodiments of the invention to store and execute target binary code;

[0010] FIG. 7 is a block diagram of an exemplary apparatus according to some embodiments of the invention to translate source binary code to target binary code and to store and execute the target binary code; and

[0011] FIG. 8 is a block diagram of yet another exemplary apparatus according to some embodiments of the invention to translate source binary code to target binary code.

[0012] It will be appreciated that for simplicity and clarity of illustration, elements shown in the figures have not necessarily been drawn to scale. For example, the dimensions of some of the elements may be exaggerated relative to other elements for clarity.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

[0013] In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of embodiments of the invention. However it will be understood by those of ordinary skill in the art that the embodiments of the invention may be practiced without these specific details. In other instances, well-known methods, procedures, components and circuits have not been described in detail so as not to obscure the embodiments of the invention.

[0014] Some portions of the detailed description which follow are presented in terms of algorithms and symbolic representations of operations on data bits or binary digital signals within a computer memory. These algorithmic descriptions and representations may be the techniques used by those skilled in the data processing arts to convey the substance of their work to others skilled in the art.

[0015] An algorithm is here, and generally, considered to be a self-consistent sequence of acts or operations leading to a desired result. These include physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers or the like. It should be understood, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

[0016] Unless specifically stated otherwise, as apparent from the following discussions, it is appreciated that throughout the specification discussions utilizing terms such as "processing," "computing," "calculating," "determining," or the like, refer to the action and/or processes of a computer or computing system, or similar electronic computing device, that manipulate and/or transform data represented as physical, such as electronic, quantities within the computing system's registers and/or memories into other data similarly represented as physical quantities within the computing system's memories, registers or other such information storage, transmission or display devices.

[0017] Embodiments of the present invention may include apparatuses for performing the operations herein. This apparatus may be specially constructed for the desired

purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), electrically programmable read-only memories (EPROMs), electrically erasable and programmable read only memories (EEPROMs), magnetic or optical cards, or any other type of media suitable for storing electronic instructions, and capable of being coupled to a computer system bus.

[0018] The processes and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct a more specialized apparatus to perform the desired method. The desired structure for a variety of these systems will appear from the description below. In addition, embodiments of the present invention are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

[0019] Fig. 1 shows a binary-code translation module 2 according to some embodiments of the invention. Binary-code translation module 2 may receive as input a binary code ("source binary code") associated with a first processor architecture having a first set of instructions ("source architecture"), and may output a binary code ("target binary code") associated with a second processor architecture having a second set of instructions ("target architecture"). The results produced by executing the target binary code on a processor that complies with the target architecture may be substantially the same as those produced by executing the source binary code on a processor that complies with the source architecture.

[0020] The instruction sets of both the source and target architectures may comply with ANSI/IEEE standard 754-1985 for binary floating-point arithmetic and may support conversion of floating point numbers to integer numbers using at least the four rounding modes defined in ANSI/IEEE standard 754-1985 to affect all arithmetic

operations except comparison and remainder: round to nearest, round toward negative infinity, round toward positive infinity, and round to zero.

[0021] It is understood by persons of ordinary skill in the art that a floating point number is implemented in a computer as a representation of a floating point number. Similarly, an integer number is implemented in a computer as a representation of a integer number. For example, 32 bits or 64 bits may be used to physically represent a floating point number, and 16 bits or 32 bits may be used to physically represent an integer number.

[0022] Prior to execution of an instruction to convert a floating point number to an integer using a specific rounding mode in the source architecture, the source architecture may need to be set to that rounding mode by, for example, a floating point control instruction.

[0023] If, for example, during execution of some portion of code, the source architecture is set to convert floating point numbers into integers using one rounding mode ("old rounding mode"), and during the execution of that code portion, a floating-point number is to be converted to integer using a second rounding mode ("new rounding mode"), the source architecture may need to execute the following exemplary sequence of four instructions (referred to as sequence "A"), :

a.} save old rounding mode
b.} execute floating point control instruction to set rounding mode to new rounding mode
c.} convert floating point to integer using new rounding mode
d.} execute floating point control instruction to set rounding mode to old rounding mode

[0024] In sequence "A", instruction a.} is followed by instruction b.}, instruction b.} is followed by instruction c.}, and instruction c.} is followed by instruction d.}. The term "followed by" means that a first instruction is executed after a second instruction, and other instructions may or may not be intervening between the second and the first instructions.

[0025] Similarly, prior to execution of an instruction to convert a floating point number to an integer using a specific rounding mode in the target architecture, the target architecture may need to be set to that rounding mode by, for example, a floating point control instruction.

[0026] If the instruction set of the target architecture comprises an instruction to round floating point numbers to floating point numbers using round to zero rounding mode, regardless of the rounding mode setting of the target architecture ("round to zero forced mode instruction"), it may be possible to avoid the use of a floating point control instruction to set the rounding mode in the target architecture.

[0027] As shown in FIGs. 2, 3, 4, 5A and 5B, binary-code translation module 2 may use the round to zero forced mode instruction of the target architecture to translate a source binary code portion comprising a sequence "A" into a target binary code portion that does not include a floating point control instruction.

[0028] FIG. 2 is a flowchart illustration according to some embodiments of the invention of an exemplary method to be implemented in a binary code translation module for translating a portion of a source binary code to a portion of a target binary code. FIGS. 3, 4, 5A and 5B are flowchart illustrations of exemplary instruction sequences to be generated by a binary code translation module, according to some embodiments of the invention.

[0029] Referring to FIG. 2, binary code translation module 2 may identify a sequence similar to exemplary sequence "A" in the source binary code (-10-) and may then identify the new rounding mode (-11-).

[0030] If the new rounding mode is round to nearest (-12-), then binary code translation module 2 may translate source binary code sequence "A" into an equivalent target binary code sequence "A" (-14-) having floating point control instructions for changing rounding modes.

[0031] If the new rounding mode is round toward negative infinity (-16-), then binary code translation module 2 may translate source binary code sequence "A" into a target binary code sequence "B" (-18-), described in FIG. 3.

[0032] If the new rounding mode is round toward positive infinity (-20-), then binary code translation module 2 may translate source binary code sequence "A" into a target binary code sequence "C" (-22-), described in FIG. 4.

[0033] If the new rounding mode is round to zero, then binary code translation module 2 may translate source binary code sequence "A" into a target binary code sequence "D" (-24-), described in FIGs. 5A and 5B.

[0034] FIG. 3 is a flowchart illustration of the method of an exemplary target binary code sequence "B", according to some embodiments of the invention. It should be noted that the method of FIG. 3 may be executed by a target processor after checking that the floating point data is a suitable floating point number. For example, floating point data defined by ANSI/IEEE standard 754-1985 for binary floating-point arithmetic as "Infinite" or as "Quiet not a number" (QNaN) may not be suitable.

[0035] A first floating point number is generated by rounding an initial floating point number using a round to zero forced mode instruction (-30-). If the first floating point number is positive (-32-) or if its value is equal to the value of the initial floating point number (-34-), then a result integer number is generated by converting the representation of the first floating point number to an integer representation (-36-). Otherwise, a second floating point number is generated by subtracting one from the first floating point number (-38-), and the result integer number is generated by converting the representation of the second floating point number to an integer representation (-39-).

[0036] FIG. 4 is a flowchart illustration of the method of an exemplary target binary code sequence "C", according to some embodiments of the invention. It should be noted that the method of FIG. 3 may be executed by a target processor after checking that the floating point data is a suitable floating point number.

[0037] A first floating point number is generated by rounding an initial floating point number using a round to zero forced mode instruction (-40-). If the first floating point number is negative (-42-) or if its value is equal to the value of the initial floating point number (-44-), then a result integer number is generated by converting the representation of the first floating point number to an integer representation (-46-). Otherwise, a second floating point number is generated by adding one to the first floating point number (-48-), and the result integer number is generated by converting the representation of the second floating point number to an integer representation (-49-).

[0038] FIG. 5A is a flowchart illustration of the operation of an exemplary target binary code sequence "D", according to some embodiments of the invention. The floating point number is converted to an integer number using the round to zero mode (-50-). FIG. 5B is a flowchart illustration of the operation of another exemplary target binary code sequence "D", according to some embodiments of the invention. The initial floating point number is converted to a first floating point number using the round to zero forced mode instruction (-52-). The result integer number is generated by converting the representation of the first floating point number to an integer representation (-54-).

[0039] FIG. 6 is a block diagram of an exemplary apparatus 60 according to some embodiments of the invention. Apparatus 60 may comprise a processor 62 and a memory 64 coupled to processor 62. Memory 64 may store a target binary code 66 generated by a binary-code translation module (not shown) using the method described in FIGs. 2 – 5. Target binary code 66 complies with the architecture of processor 62. Processor 62 may execute target binary code 66.

[0040] FIG. 7 is a block diagram of an exemplary apparatus 70 according to some embodiments of the invention. Apparatus 70 may comprise a processor 72 and a memory 74 coupled to processor 72. Memory 74 may store a source binary code 76 that does not comply with the architecture of processor 72, and a binary code translation module 78, suitable to translate source binary code 76 to a target binary code that complies with the architecture of processor 72. Processor 72 may execute binary code translation module 78 to translate source binary code 76 into a target binary code 80. Target binary code 80 may then be stored in memory 74 and may be executed by processor 72. Alternatively, source binary code 76, binary-code translation module 78 and target binary code 80 may be stored in separate memories that are coupled to processor 72.

[0041] FIG. 8 is a block diagram of an exemplary apparatus according to some embodiments of the invention to translate source binary code to target binary code.

[0042] Apparatus 90 may comprise a processor 92 and a memory 94 coupled to processor 92.

[0043] Memory 94 may store a binary code translation module 96. Binary code translation module 96 may be able to translate a source binary code associated with a

source processor architecture into a target binary code associated with a target processor architecture. The architecture of processor 92 may, or may not, comply with the source architecture and may, or may not, comply with the target architecture.

[0044] Processor 92 may receive source binary code, may execute binary code translation module 96, and may output target binary code.

[0045] A non-exhaustive list of examples for apparatuses 60, 70 and 90 includes a desktop personal computer, a work station, a server computer, a laptop computer, a notebook computer, a hand-held computer, a personal digital assistant (PDA), a mobile telephone, and the like.

[0046] A non-exhaustive list of examples for processors 62, 72 and 92 includes a central processing unit (CPU), a digital signal processor (DSP), a reduced instruction set computer (RISC), a complex instruction set computer (CISC) and the like. Moreover, processors 62, 72 and 92 may be part of an application specific integrated circuit (ASIC) or may be a part of an application specific standard product (ASSP).

[0047] Memories 64, 74 and 94 may be fixed in or removable from apparatuses 60, 70 and 90, respectively. A non-exhaustive list of examples for memories 64, 74 and 94 includes any combination of the followings:

- semiconductor devices, such as
  - synchronous dynamic random access memory (SDRAM) devices, RAMBUS dynamic random access memory (RDRAM) devices, double data rate (DDR) memory devices, static random access memory (SRAM), flash memory devices, electrically erasable programmable read only memory devices (EEPROM), non-volatile random access memory devices (NVRAM), universal serial bus (USB) removable memory, and the like,
- optical devices, such as
  - compact disk read only memory (CD ROM), and the like,
- and magnetic devices, such as
  - a hard disk, a floppy disk, a magnetic tape, and the like.

[0048] While certain features of the invention have been illustrated and described herein, many modifications, substitutions, changes, and equivalents will now occur to those of ordinary skill in the art. It is, therefore, to be understood that the appended

claims are intended to cover all such modifications and changes as fall within the spirit of the invention.